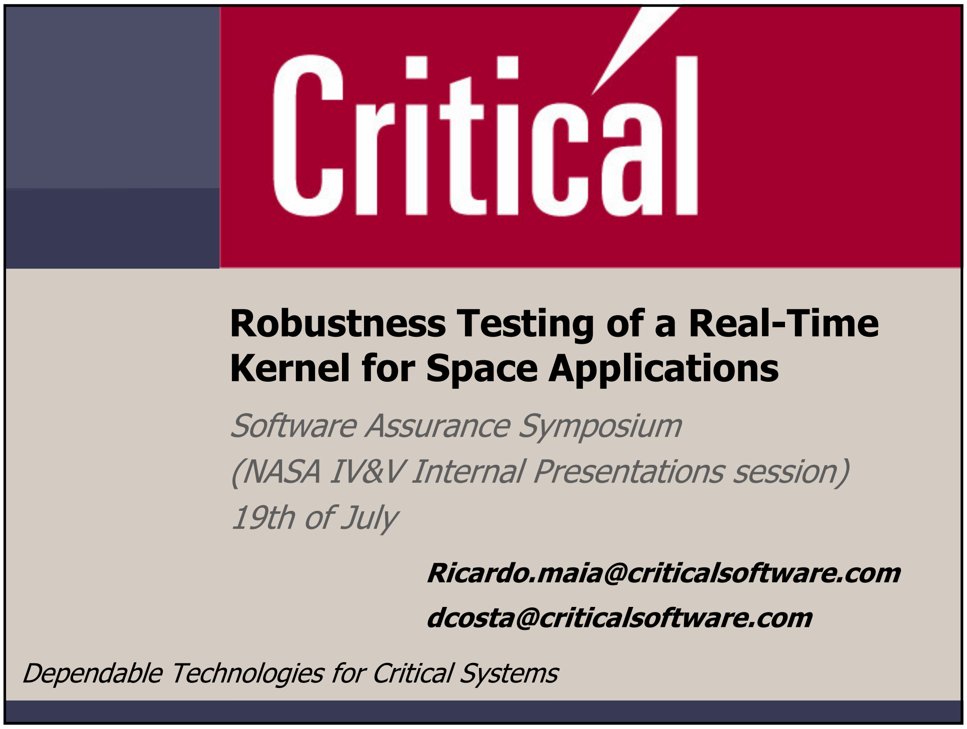


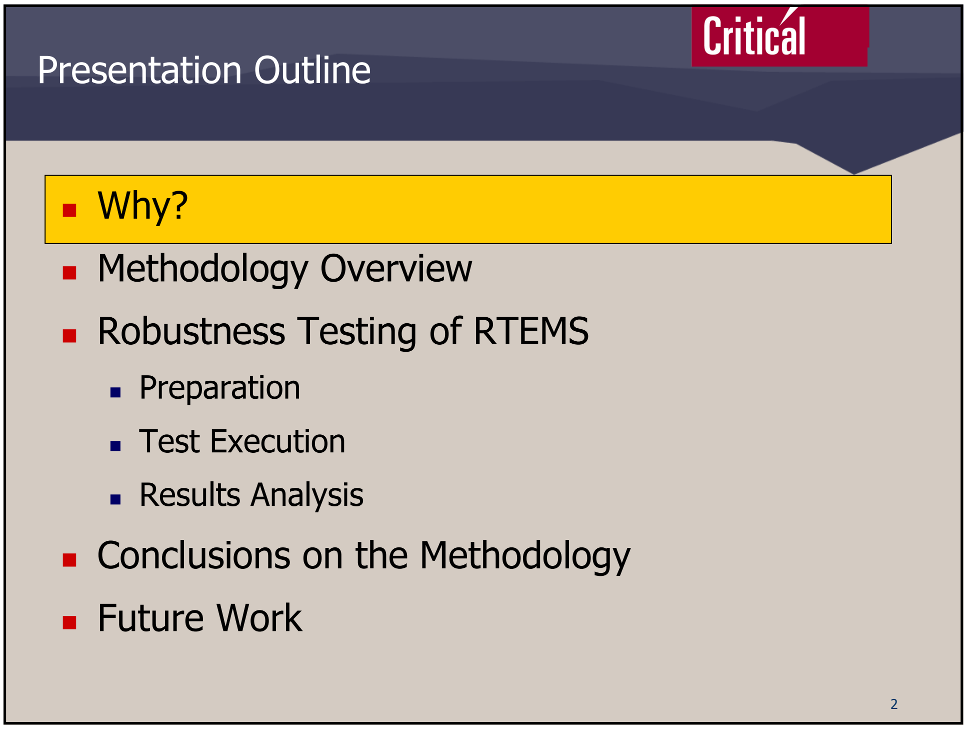
Critical

Robustness Testing of a Real-Time Kernel for Space Applications

Software Assurance Symposium
(NASA IV&V Internal Presentations session)
19th of July

Ricardo.maia@criticalsoftware.com
dcosta@criticalsoftware.com

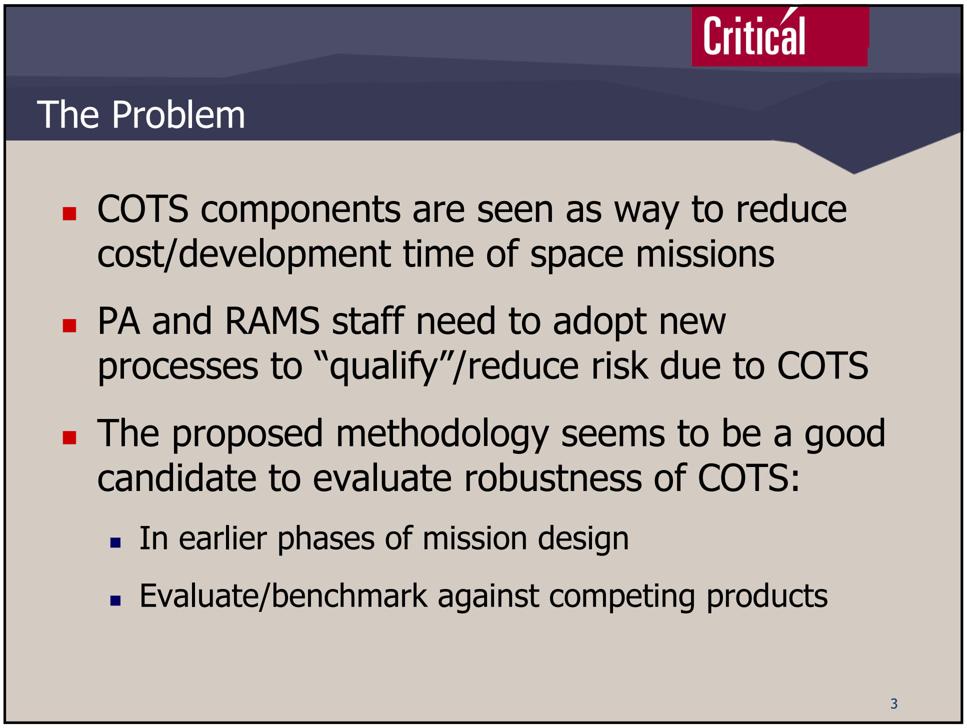
Dependable Technologies for Critical Systems



Critical

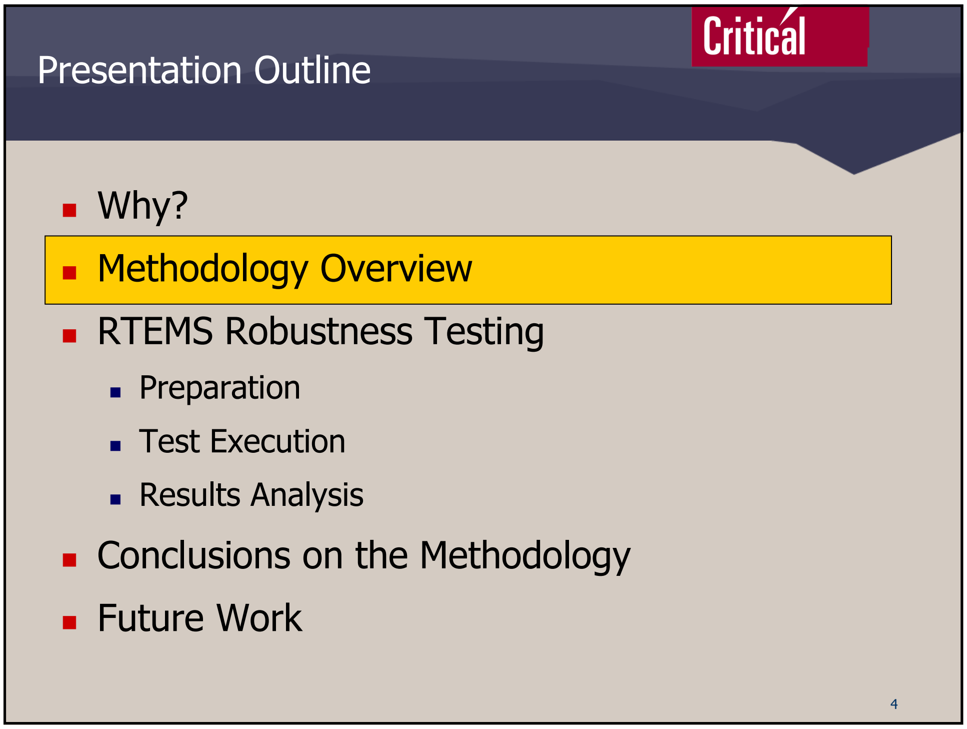
Presentation Outline

- Why?
- Methodology Overview
- Robustness Testing of RTEMS
  - Preparation
  - Test Execution
  - Results Analysis
- Conclusions on the Methodology
- Future Work



The Problem

- COTS components are seen as way to reduce cost/development time of space missions
- PA and RAMS staff need to adopt new processes to “qualify”/reduce risk due to COTS
- The proposed methodology seems to be a good candidate to evaluate robustness of COTS:
  - In earlier phases of mission design
  - Evaluate/benchmark against competing products



Presentation Outline

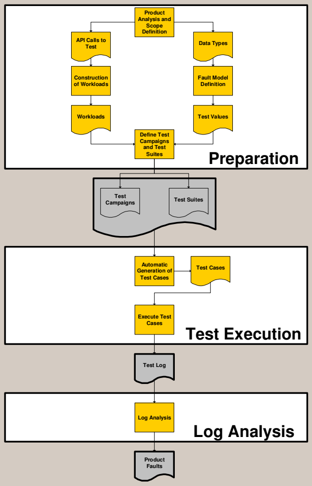
- Why?
- Methodology Overview
- RTEMS Robustness Testing
  - Preparation
  - Test Execution
  - Results Analysis
- Conclusions on the Methodology
- Future Work

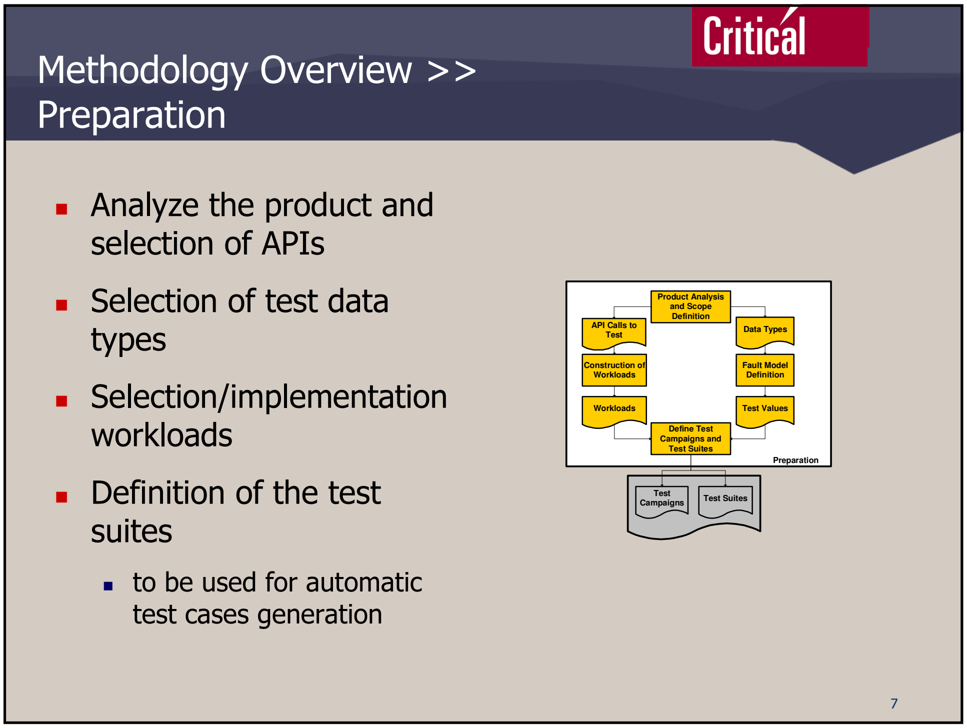
Methodology Overview >> Goal

- COTS testing under a non-nominal profile
  - exercising consistency checking and error handling mechanisms
- Use non-nominal parameters in system calls
  - e.g. provide a NULL pointer instead of a pointer to a memory region
- Check if COTS behavior conforms to its specification

Methodology Overview >> Phases

- It comprises 3 phases:
  - Preparation
  - Test Execution
  - Log Analysis





Methodology Overview >> Preparation

- Analyze the product and selection of APIs
- Selection of test data types
- Selection/implementation workloads
- Definition of the test suites
    - to be used for automatic test cases generation

Product Analysis and Scope Definition

API Calls to Test

Data Types

Construction of Workloads

Fault Model Definition

Workloads

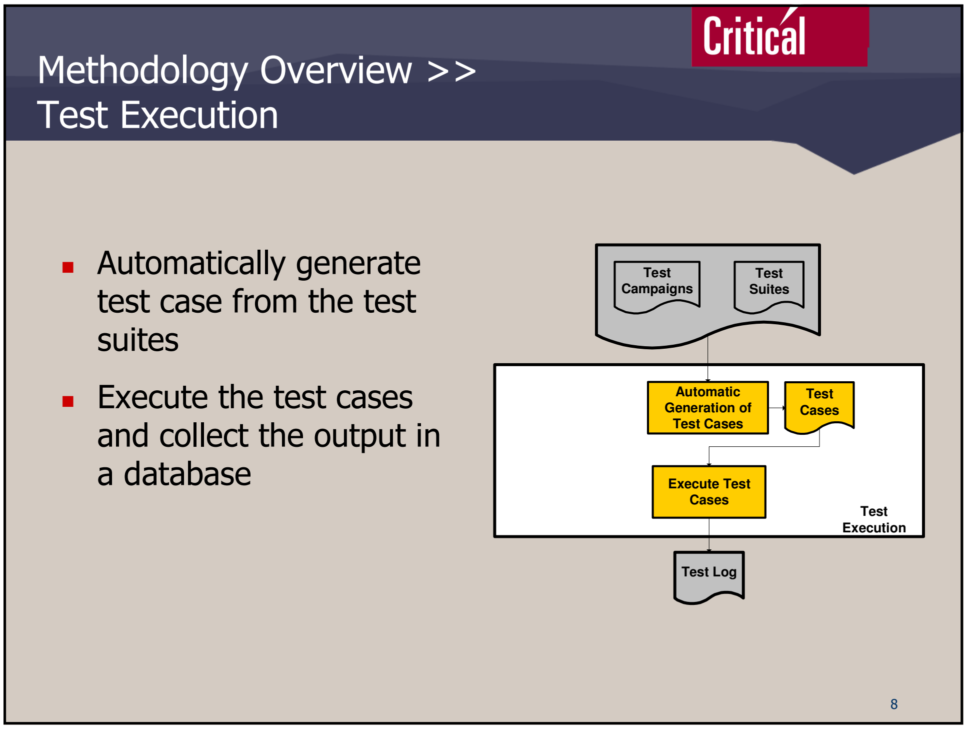
Test Values

Define Test Campaigns and Test Suites

Preparation

Test Campaigns

Test Suites



Methodology Overview >> Test Execution

- Automatically generate test case from the test suites
- Execute the test cases and collect the output in a database

Test Campaigns

Test Suites

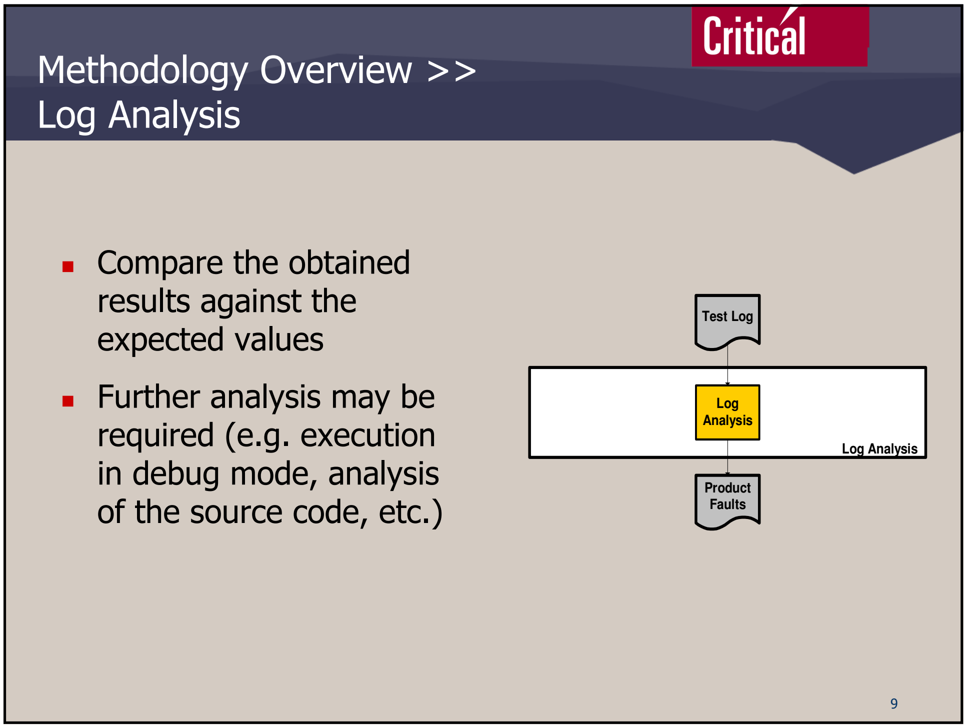
Automatic Generation of Test Cases

Test Cases

Execute Test Cases

Test Execution

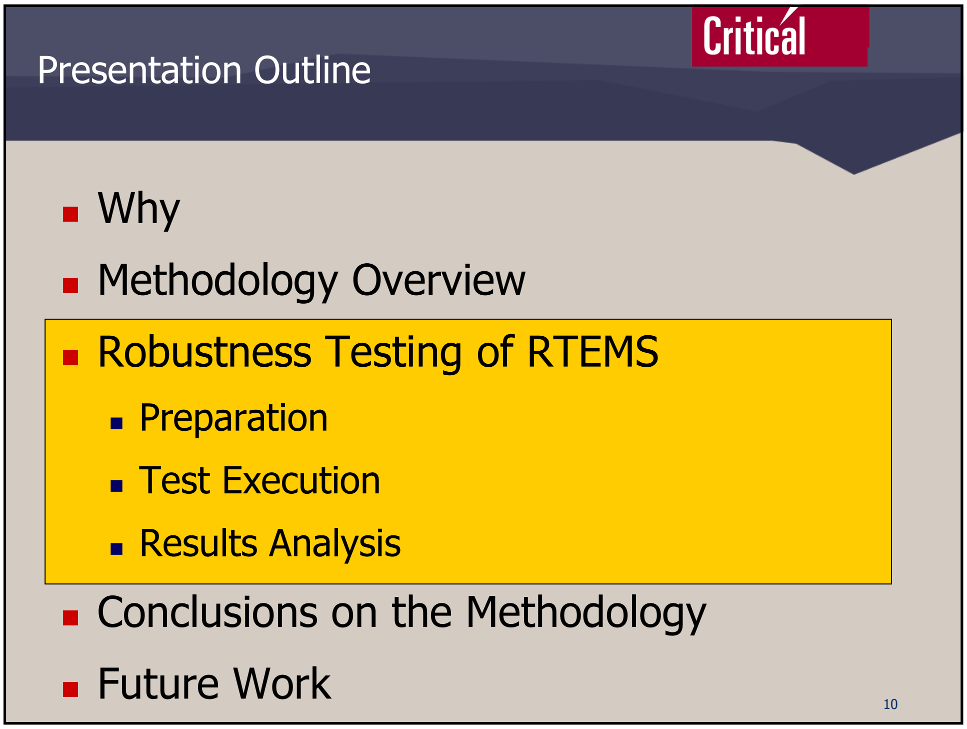
Test Log



Methodology Overview >> Log Analysis

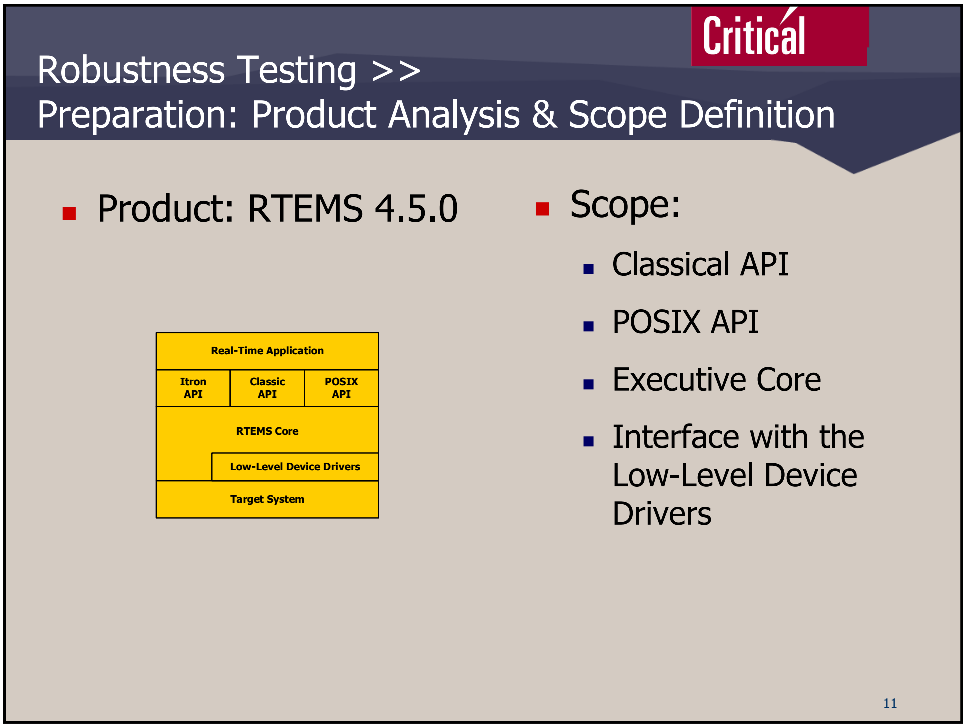
- Compare the obtained results against the expected values
- Further analysis may be required (e.g. execution in debug mode, analysis of the source code, etc.)

Test Log
Log Analysis
Log Analysis
Product Faults



Presentation Outline

- Why
- Methodology Overview
- Robustness Testing of RTEMS
  - Preparation
  - Test Execution
  - Results Analysis
- Conclusions on the Methodology
- Future Work

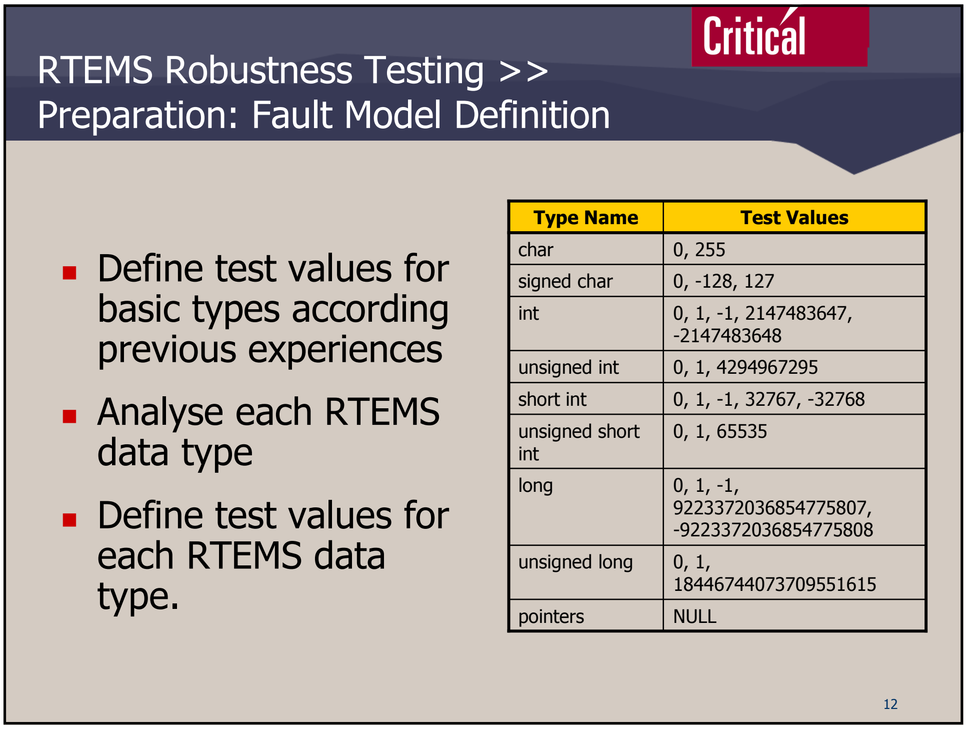


Robustness Testing >> Preparation: Product Analysis & Scope Definition

- Product: RTEMS 4.5.0

Real-Time Application
Itron API
Classic API
POSIX API
RTEMS Core
Low-Level Device Drivers
Target System

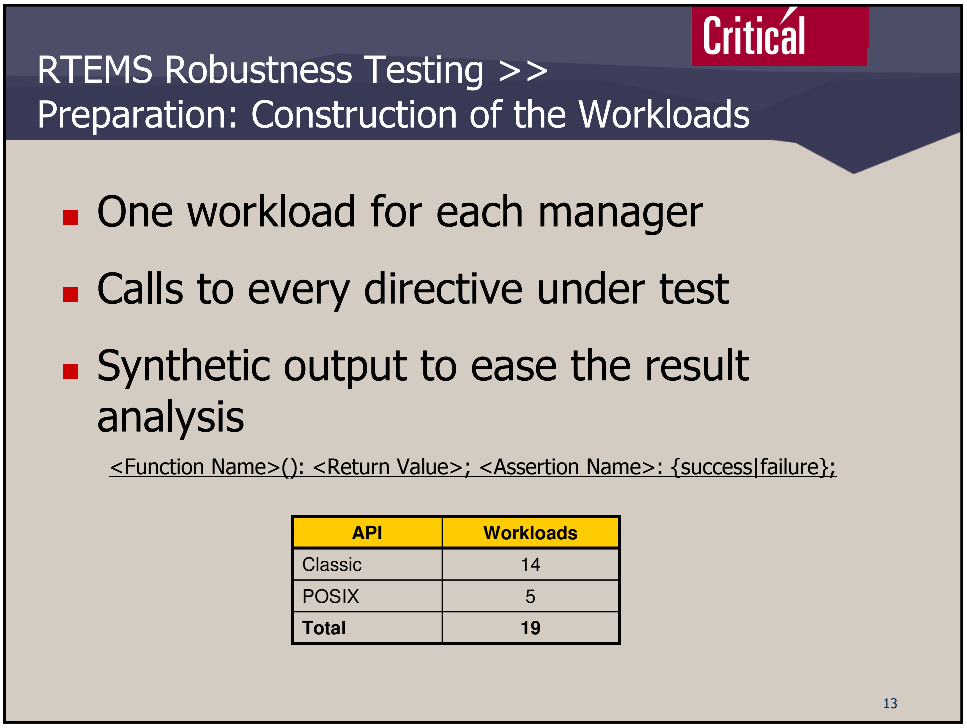
- Scope:
  - Classical API
  - POSIX API
  - Executive Core
  - Interface with the Low-Level Device Drivers



RTEMS Robustness Testing >> Preparation: Fault Model Definition

- Define test values for basic types according previous experiences
- Analyse each RTEMS data type
- Define test values for each RTEMS data type.

Type Name	Test Values
char	0, 255
signed char	0, -128, 127
int	0, 1, -1, 2147483647, -2147483648
unsigned int	0, 1, 4294967295
short int	0, 1, -1, 32767, -32768
unsigned short int	0, 1, 65535
long	0, 1, -1, 9223372036854775807, -9223372036854775808
unsigned long	0, 1, 18446744073709551615
pointers	NULL

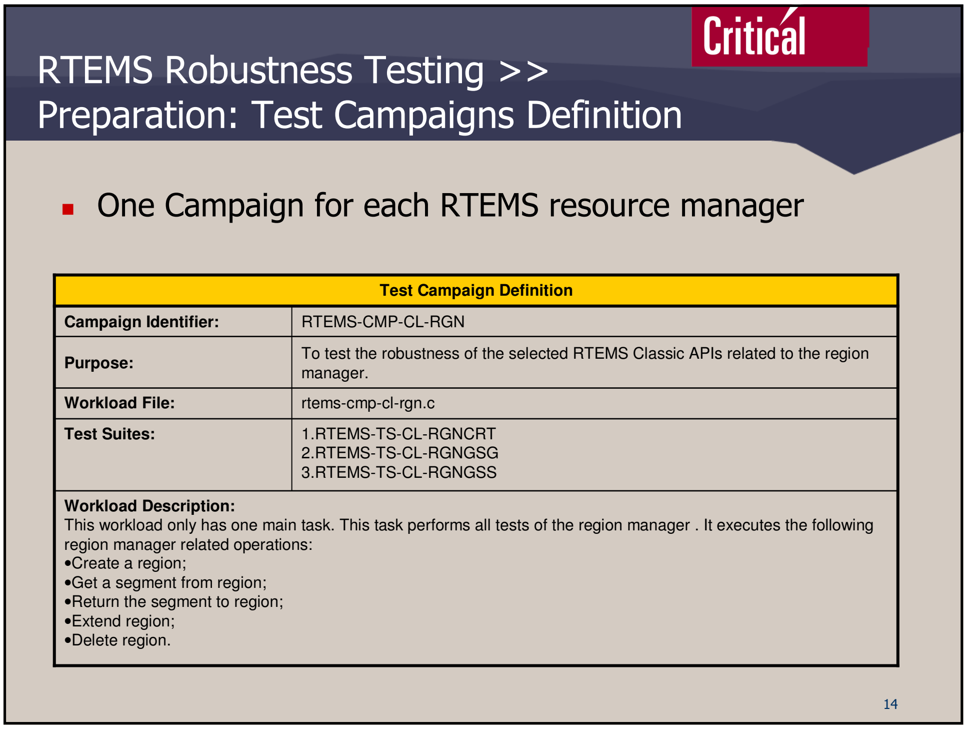


RTEMS Robustness Testing >> Preparation: Construction of the Workloads

- One workload for each manager
- Calls to every directive under test
- Synthetic output to ease the result analysis

<Function Name>(): <Return Value>; <Assertion Name>: {success|failure};

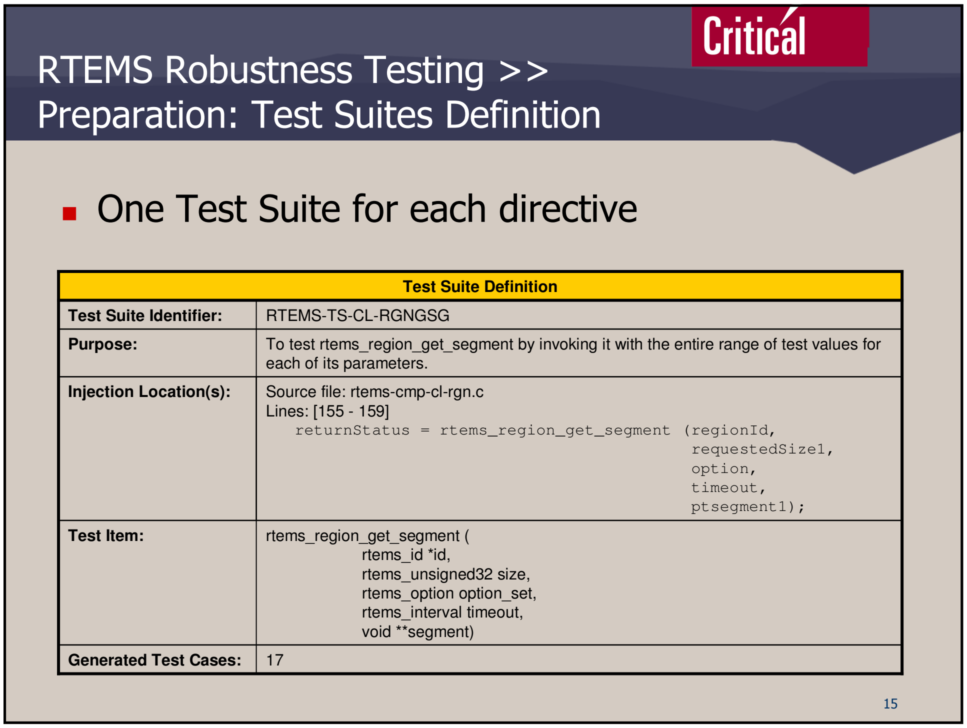
API	Workloads
Classic	14
POSIX	5
Total	**19**



RTEMS Robustness Testing >> Preparation: Test Campaigns Definition

- One Campaign for each RTEMS resource manager

Test Campaign Definition	
Campaign Identifier:	RTEMS-CMP-CL-RGN
Purpose:	To test the robustness of the selected RTEMS Classic APIs related to the region manager.
Workload File:	rtems-cmp-cl-rgn.c
Test Suites:	1.RTEMS-TS-CL-RGNCRT 2.RTEMS-TS-CL-RGNGSG 3.RTEMS-TS-CL-RGNGSS
Workload Description: This workload only has one main task. This task performs all tests of the region manager . It executes the following region manager related operations: •Create a region; •Get a segment from region; •Return the segment to region; •Extend region; •Delete region.	



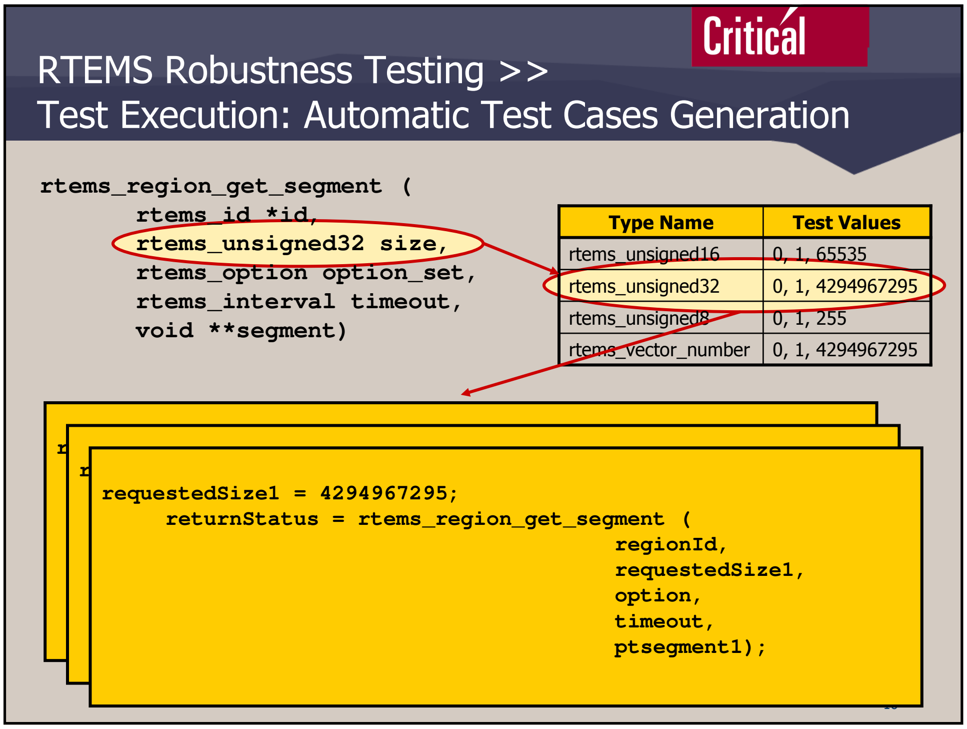
Critical

RTEMS Robustness Testing >> Preparation: Test Suites Definition

- One Test Suite for each directive

Test Suite Definition	
Test Suite Identifier:	RTEMS-TS-CL-RGNGSG
Purpose:	To test rtems_region_get_segment by invoking it with the entire range of test values for each of its parameters.
Injection Location(s):	Source file: rtems-cmp-cl-rgn.c Lines: [155 - 159] `returnStatus = rtems_region_get_segment (regionId, requestedSize1, option, timeout, ptsegment1);`
Test Item:	rtems_region_get_segment (rtems_id *id, rtems_unsigned32 size, rtems_option option_set, rtems_interval timeout, void **segment)
Generated Test Cases:	17

15



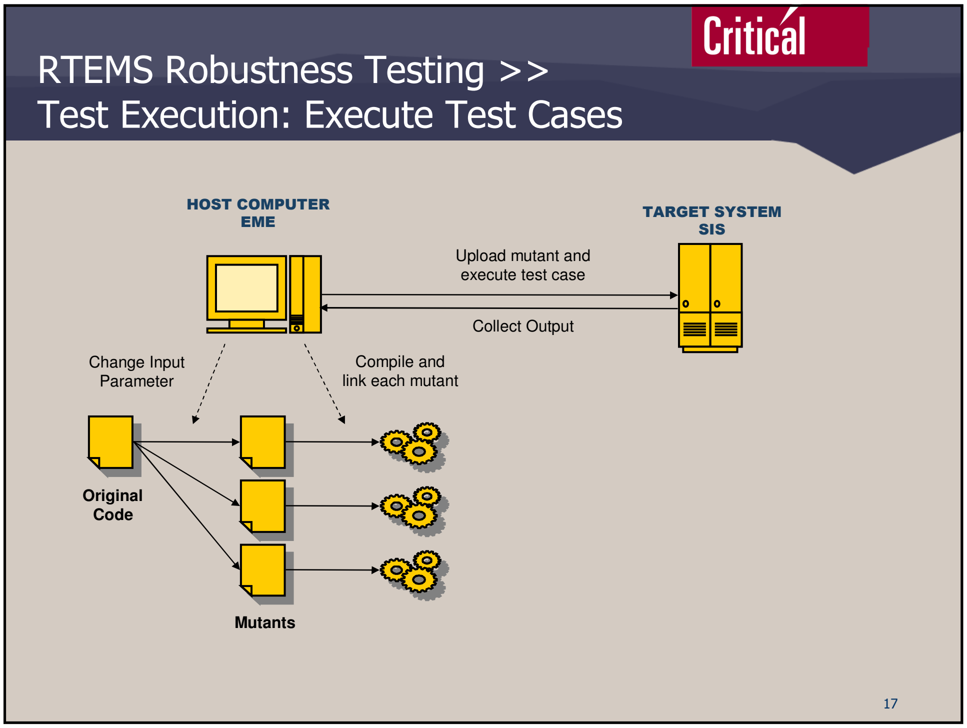
Critical

RTEMS Robustness Testing >> Test Execution: Automatic Test Cases Generation

```
rtems_region_get_segment (
       rtems_id *id,
       rtems_unsigned32 size,
       rtems_option option_set,
       rtems_interval timeout,
       void **segment)
```

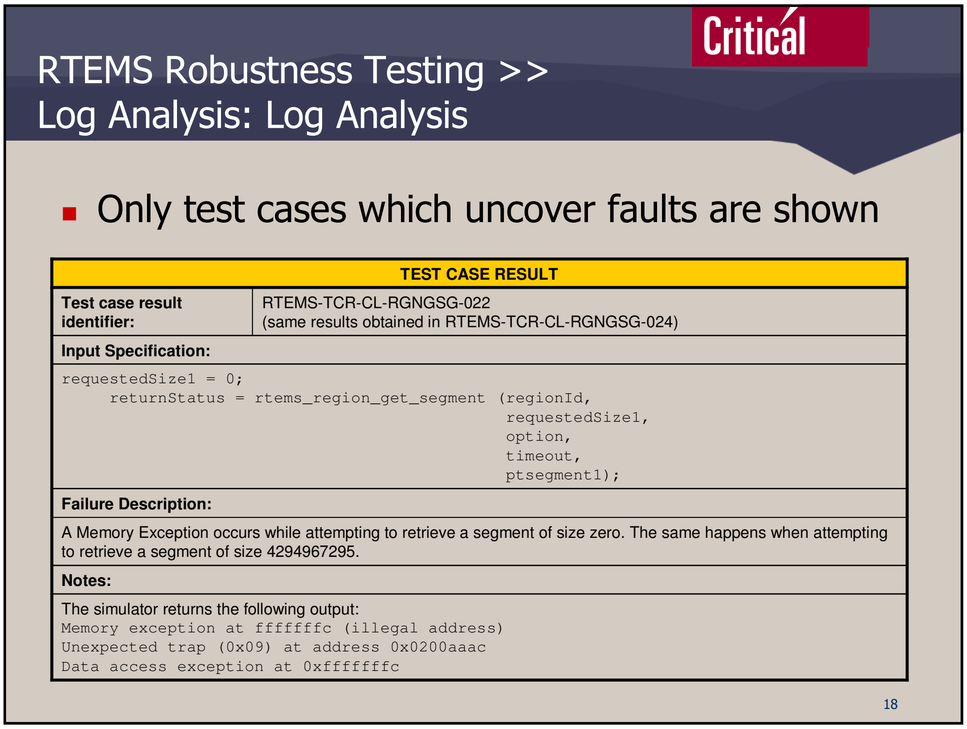
Type Name	Test Values
rtems_unsigned16	0, 1, 65535
rtems_unsigned32	0, 1, 4294967295
rtems_unsigned8	0, 1, 255
rtems_vector_number	0, 1, 4294967295

```
requestedSize1 = 4294967295;
     returnStatus = rtems_region_get_segment (
                                   regionId,
                                   requestedSize1,
                                   option,
                                   timeout,
                                   ptsegment1);
```



Critical

RTEMS Robustness Testing >> Test Execution: Execute Test Cases



Critical

RTEMS Robustness Testing >> Log Analysis: Log Analysis

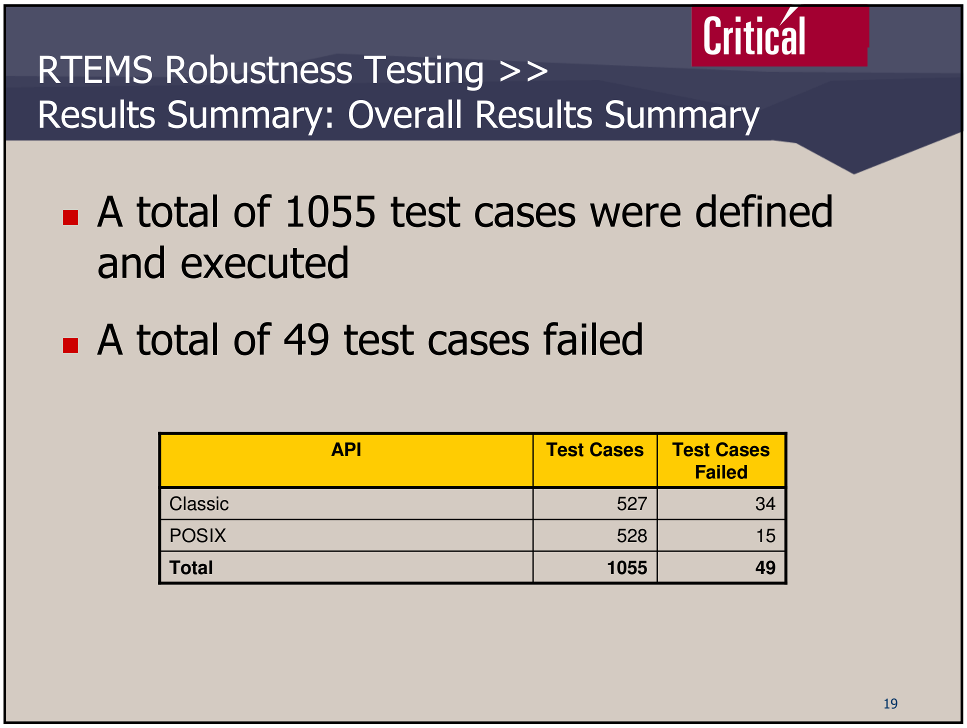
- Only test cases which uncover faults are shown

TEST CASE RESULT	
Test case result identifier:	RTEMS-TCR-CL-RGNGSG-022 (same results obtained in RTEMS-TCR-CL-RGNGSG-024)
Input Specification:	

`requestedSize1 = 0;`
`returnStatus = rtems_region_get_segment (regionId,`
`requestedSize1,`
`option,`
`timeout,`
`ptsegment1);`	

Failure Description:	
A Memory Exception occurs while attempting to retrieve a segment of size zero. The same happens when attempting to retrieve a segment of size 4294967295.	
Notes:	

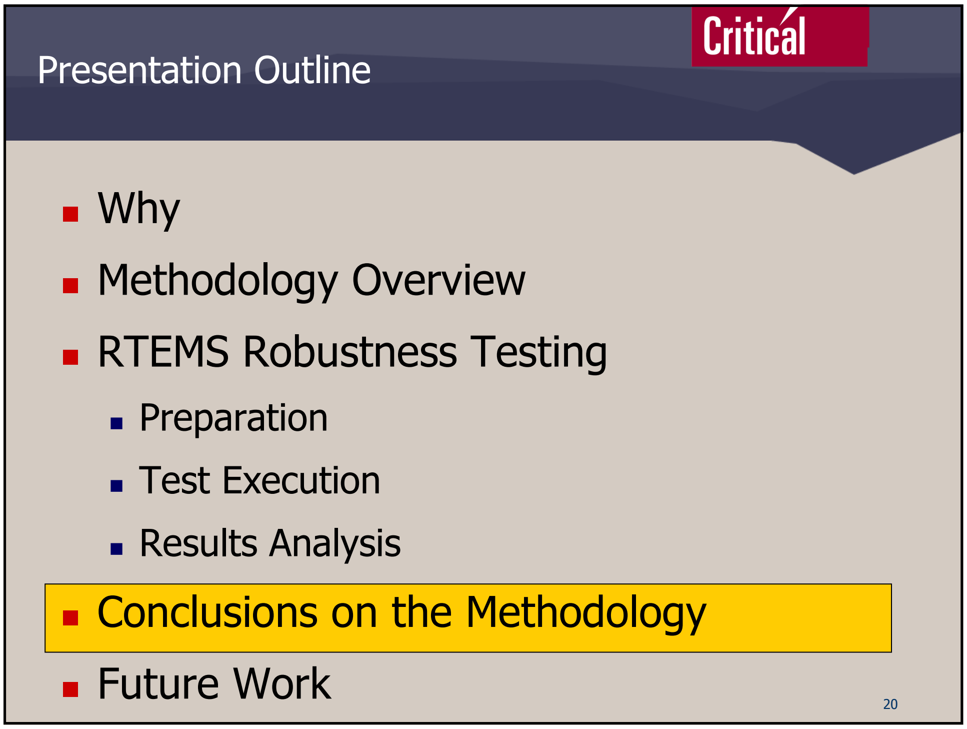
The simulator returns the following output:
`Memory exception at fffffffc (illegal address)`
`Unexpected trap (0x09) at address 0x0200aaac`
`Data access exception at 0xfffffffc`	



RTEMS Robustness Testing >> Results Summary: Overall Results Summary

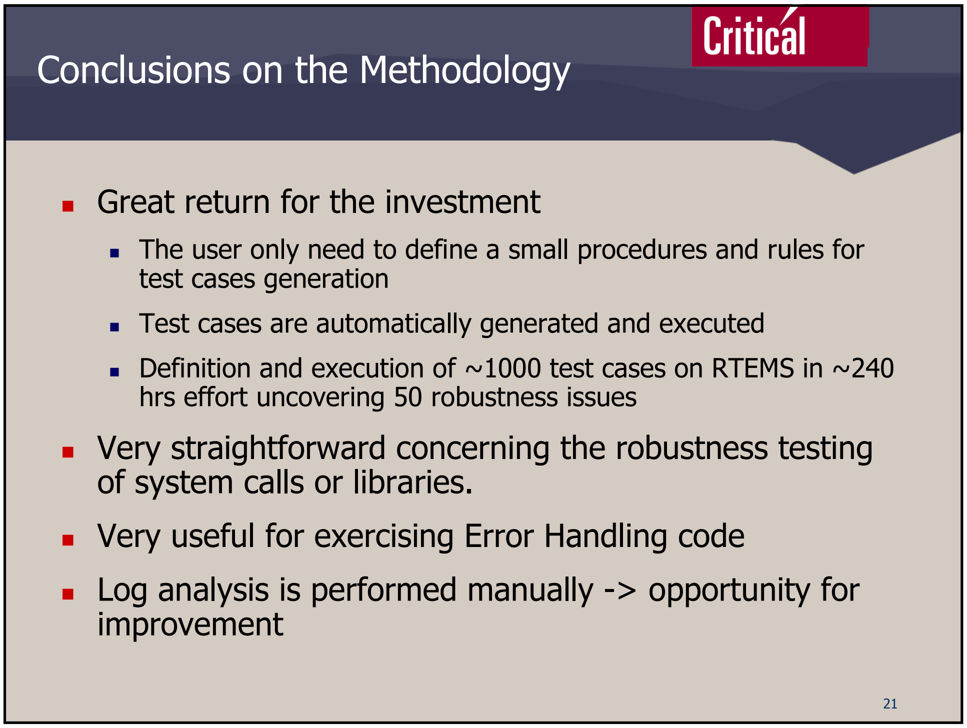
- A total of 1055 test cases were defined and executed
- A total of 49 test cases failed

API	Test Cases	Test Cases Failed
Classic	527	34
POSIX	528	15
Total	**1055**	**49**



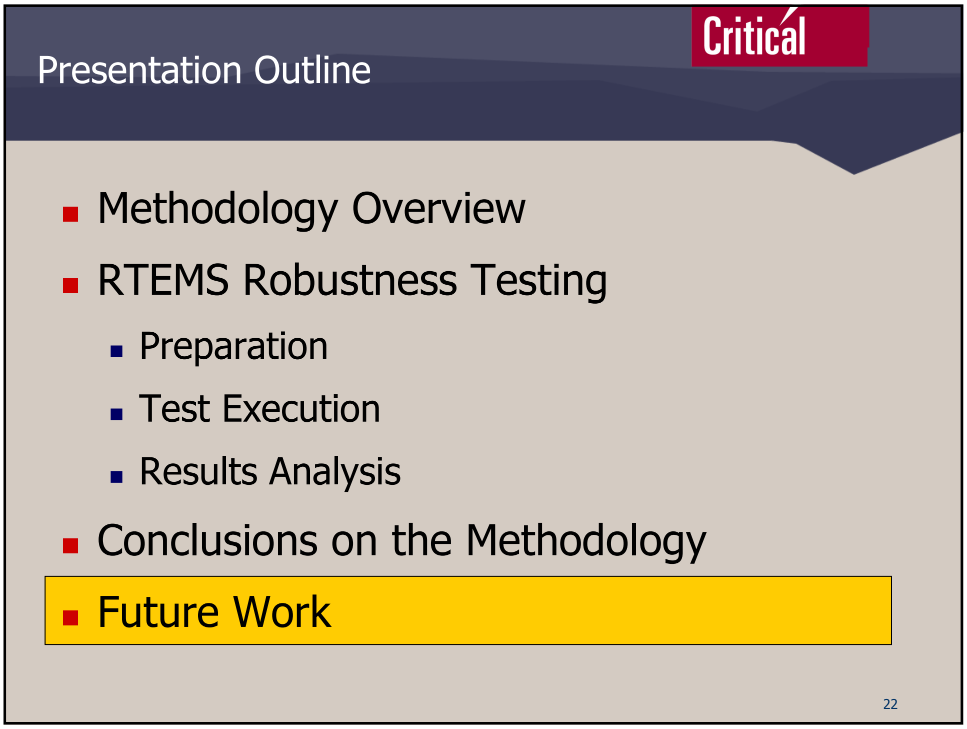
Presentation Outline

- Why
- Methodology Overview
- RTEMS Robustness Testing
  - Preparation
  - Test Execution
  - Results Analysis
- Conclusions on the Methodology
- Future Work



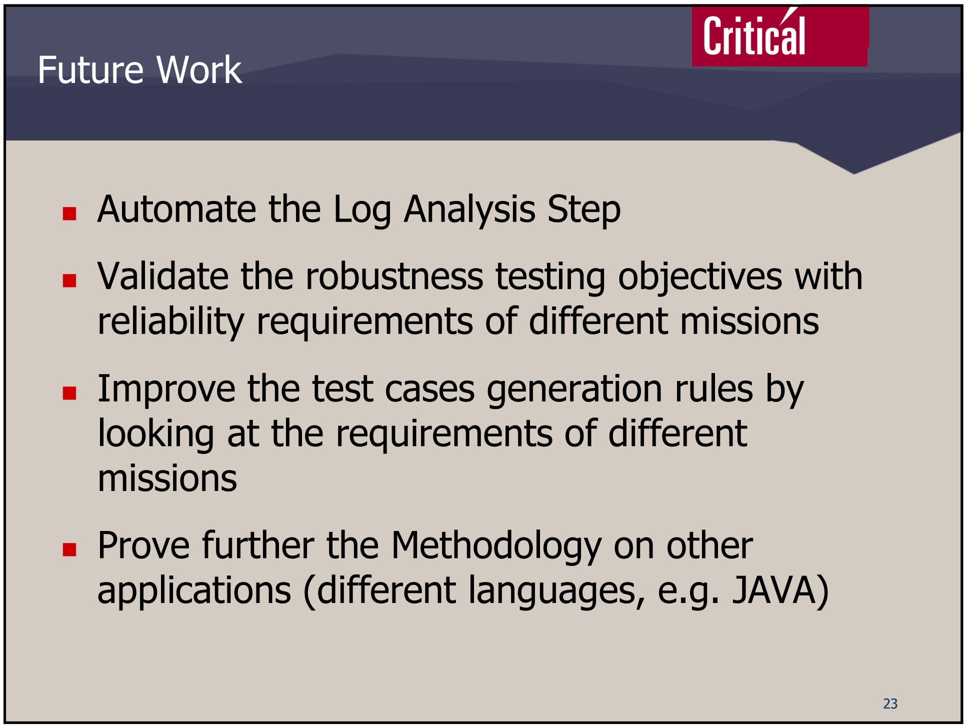
Conclusions on the Methodology

- Great return for the investment
  - The user only need to define a small procedures and rules for test cases generation
  - Test cases are automatically generated and executed
  - Definition and execution of ~1000 test cases on RTEMS in ~240 hrs effort uncovering 50 robustness issues
- Very straightforward concerning the robustness testing of system calls or libraries.
- Very useful for exercising Error Handling code
- Log analysis is performed manually -> opportunity for improvement



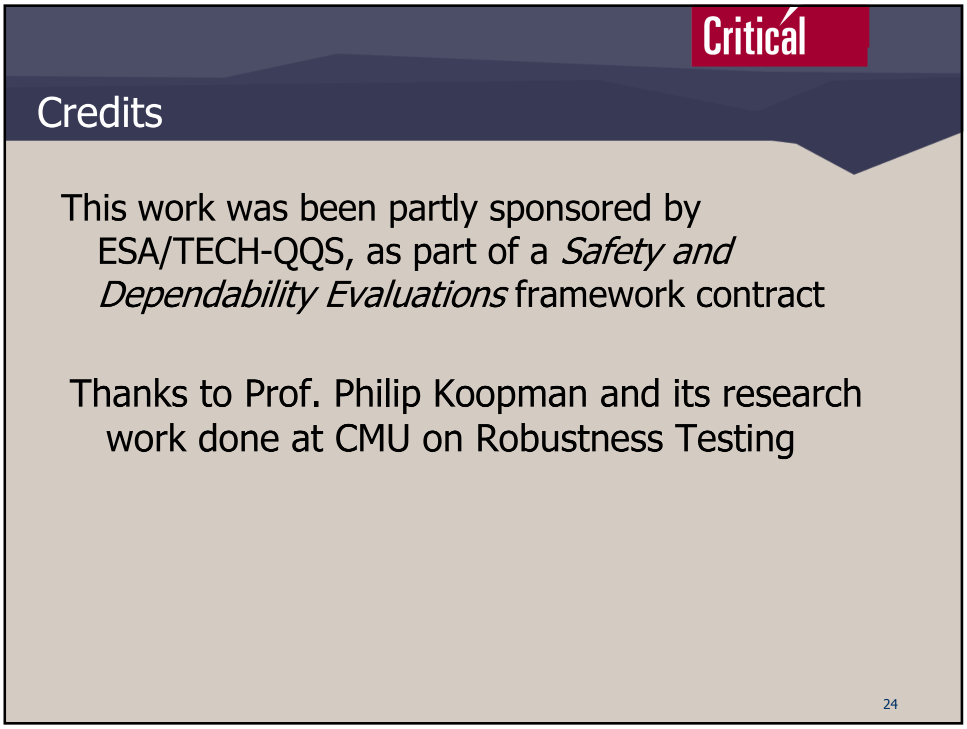
Presentation Outline

- Methodology Overview
- RTEMS Robustness Testing
  - Preparation
  - Test Execution
  - Results Analysis
- Conclusions on the Methodology
- Future Work



Future Work

- Automate the Log Analysis Step
- Validate the robustness testing objectives with reliability requirements of different missions
- Improve the test cases generation rules by looking at the requirements of different missions
- Prove further the Methodology on other applications (different languages, e.g. JAVA)



Credits

This work was been partly sponsored by ESA/TECH-QQS, as part of a *Safety and Dependability Evaluations* framework contract

Thanks to Prof. Philip Koopman and its research work done at CMU on Robustness Testing

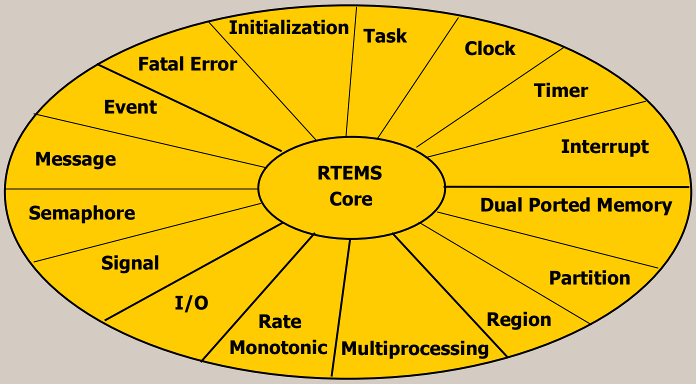
Critical

Backup Slides

Dependable Technologies for Critical Systems

Critical

RTEMS Robustness Testing >> Preparation: RTEMS Classic API



RTEMS Core

Initialization

Task

Clock

Timer

Interrupt

Dual Ported Memory

Partition

Region

Multiprocessing

Rate Monotonic

I/O

Signal

Semaphore

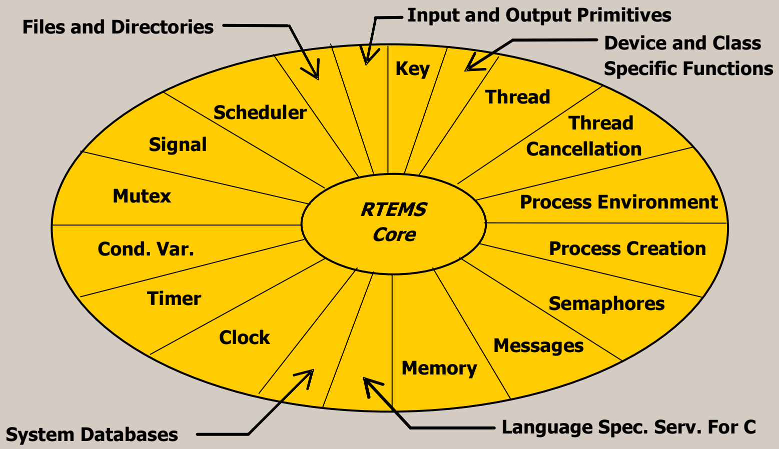
Message

Event

Fatal Error

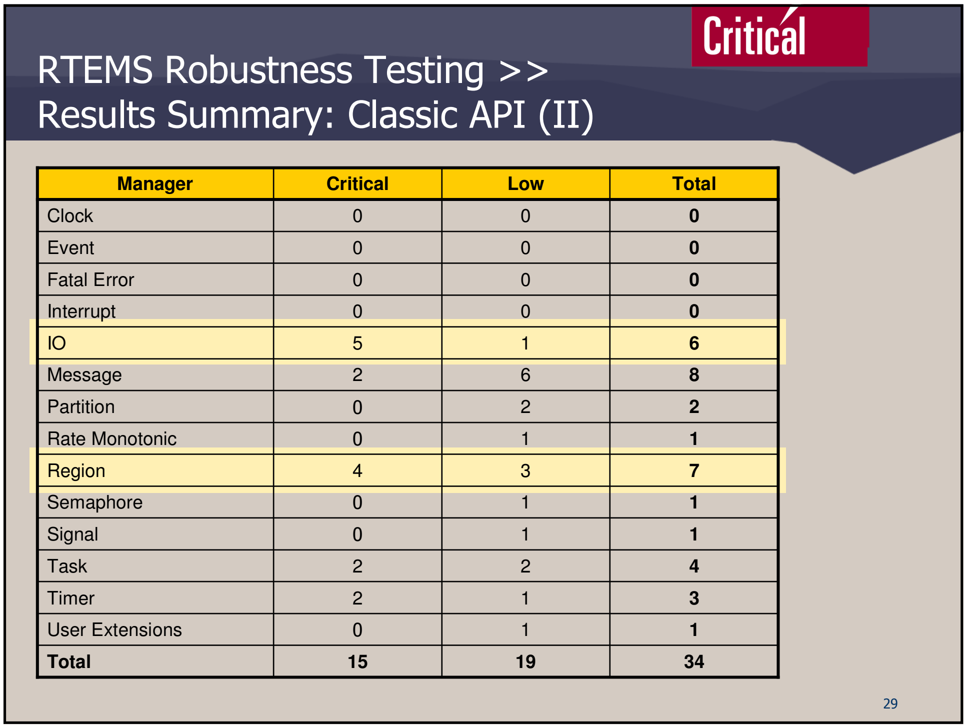
- ~100 Directives (e.g. System Calls)

RTEMS Robustness Testing >> Preparation: RTEMS POSIX API



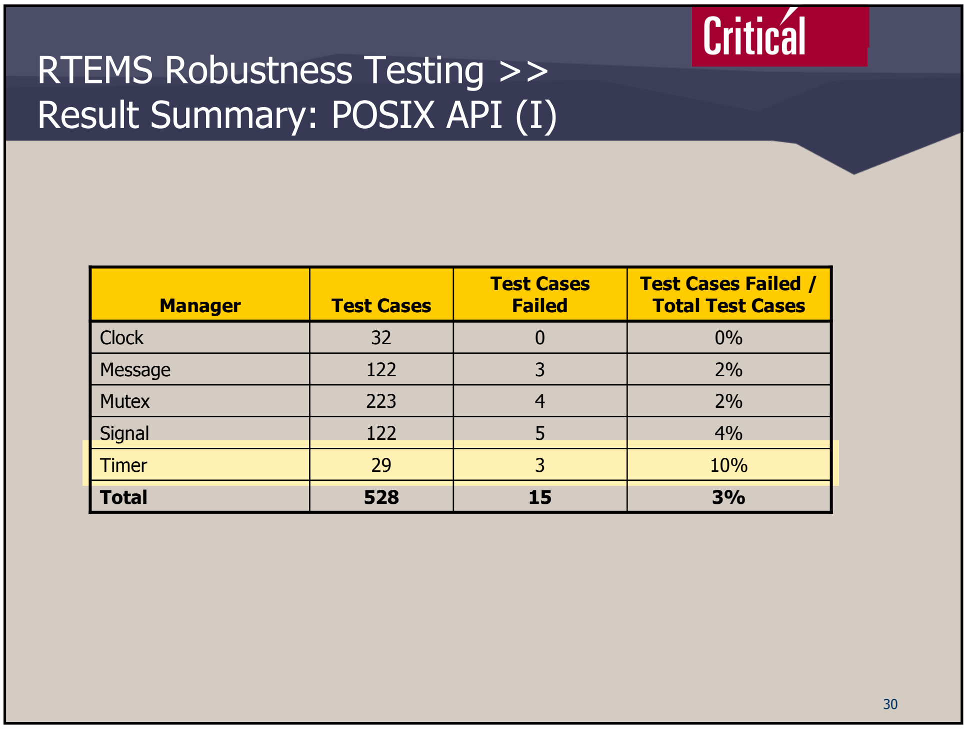
RTEMS Robustness Testing >> Results Summary: Classic API (I)

Manager	Test Cases	Test Cases Failed	Test cases Failed / Total Test Cases
Clock	68	0	0%
Event	18	0	0%
Fatal Error	3	0	0%
Interrupt	5	0	0%
IO	50	6	12%
Message	83	8	10%
Partition	27	2	7%
Rate Monotonic	24	1	4%
Region	67	7	10%
Semaphore	33	1	3%
Signal	10	1	10%
Task	55	4	7%
Timer	67	3	4%
User Extensions	17	0	0%
Total	**527**	**33**	**6%**



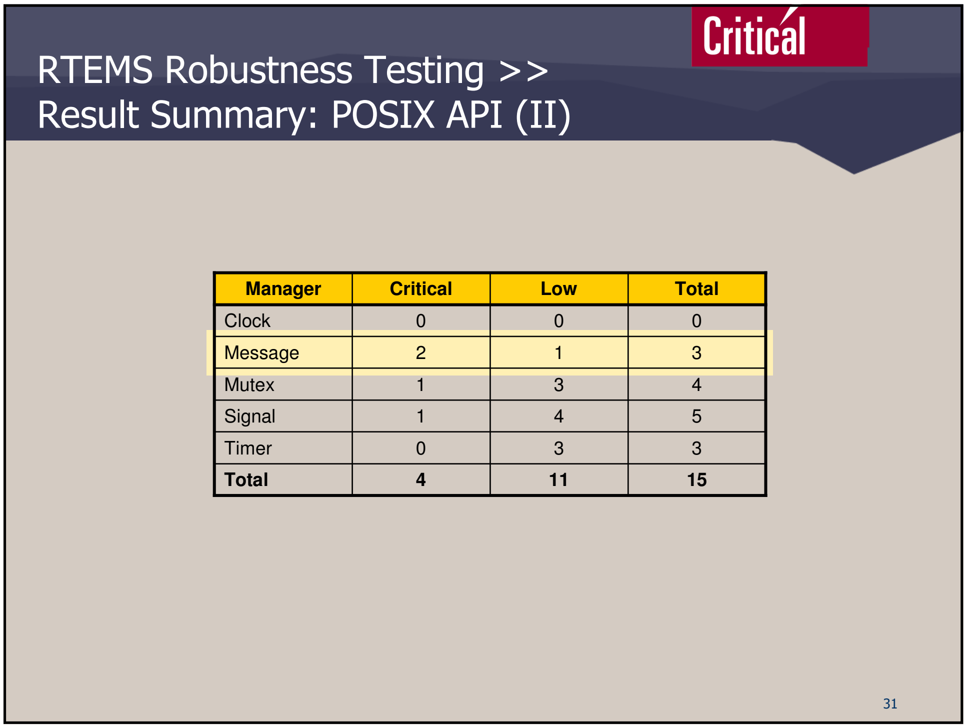
RTEMS Robustness Testing >> Results Summary: Classic API (II)

Manager	Critical	Low	Total
Clock	0	0	**0**
Event	0	0	**0**
Fatal Error	0	0	**0**
Interrupt	0	0	**0**
IO	5	1	**6**
Message	2	6	**8**
Partition	0	2	**2**
Rate Monotonic	0	1	**1**
Region	4	3	**7**
Semaphore	0	1	**1**
Signal	0	1	**1**
Task	2	2	**4**
Timer	2	1	**3**
User Extensions	0	1	**1**
Total	**15**	**19**	**34**



RTEMS Robustness Testing >> Result Summary: POSIX API (I)

Manager	Test Cases	Test Cases Failed	Test Cases Failed / Total Test Cases
Clock	32	0	0%
Message	122	3	2%
Mutex	223	4	2%
Signal	122	5	4%
Timer	29	3	10%
Total	**528**	**15**	**3%**



RTEMS Robustness Testing >> Result Summary: POSIX API (II)

Manager	Critical	Low	Total
Clock	0	0	0
Message	2	1	3
Mutex	1	3	4
Signal	1	4	5
Timer	0	3	3
Total	**4**	**11**	**15**